6
Dependency Parsing

Dependency parsing is to identify a rooted tree underlying a sentence. The nodes in this tree are the sentence tokens. The dependency tree represents the syntactic dependencies among the sentence tokens.

6.1
Task Formalization

Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_N)$ be a sentence, where x_i is the i-th token and x_0 is an artificial token which is always the root of the dependency tree. For a given input sentence $\boldsymbol{x}$, the prediction output space $\mathcal{Y}(\boldsymbol{x})$ is the set of all rooted trees whose nodes are the tokens in $\boldsymbol{x}$ and the root node is x_0. For any dependency tree $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$, we say that $(i, j) \in \boldsymbol{y}$ whenever token x_j *modifies* the *head* token x_i in the tree $\boldsymbol{y}$.

6.2
Feature Factorization

We follow McDonald et al. (2005, 2006); McDonald and Pereira (2006) and factorize the joint feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$ along the edges of the dependency tree $\boldsymbol{y}$. In that way, an edge (i, j) connecting x_i to x_j is represented by a vector $\boldsymbol{\Phi}(\boldsymbol{x}, i, j) = (\phi_1(\boldsymbol{x}, i, j), \ldots, \phi_M(\boldsymbol{x}, i, j))$ of M binary features. These features describe the dependency between the head token x_i and the modifier token x_j. Then, the *global* feature vector is

$$\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j) \in \boldsymbol{y}} \boldsymbol{\Phi}(\boldsymbol{x}, i, j),$$

which gives the frequency distribution of the local features in the tree $\boldsymbol{y}$.

6.3
Prediction Problem

The prediction problem for this DP modeling is reduced to the maximum branching problem, which can be efficiently solved by Chu-Liu-Edmonds algorithm. In the following, we just summarize this result.

The objective function of the prediction problem is

$$s(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle.$$

Using the aforementioned factorization, it is easy to see that

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j) \in \boldsymbol{y}} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, i, j) \rangle,$$

which is just the sum of the edge weights, that is,

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j) \in \boldsymbol{y}} s(i, j),$$

where $s(i, j) = \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, i, j) \rangle$ gives the weight of edge (i, j). The objective function of the prediction problem is equivalent to the tree weight given by this function. Thus, we can generate a maximum branching instance using $s(i, j)$ as edge weight function. In that way, to solve this instance is equivalent to solve the DP prediction problem in the ESL framework.

We use a loss function that just counts how many predicted edges are not correct, that is $\ell(\boldsymbol{y}, \boldsymbol{y}') = \sum_{(i,j) \in \boldsymbol{y}'} \mathbf{1}[(i, j) \notin \boldsymbol{y}]$.

6.4 Basic Features

We use the same basic features proposed by McDonald et al. (2006). For a given edge (i, j), we have the following feature list:

- *Side* – Whether x_j is on the left or on the right side of x_i in the input sentence.
- *Distance* – How many tokens there are between x_i and x_j.
- *Word* – Surface representation of both x_i and x_j.
- *POS* – Part-of-speech tag of x_i, x_{i-1}, x_{i+1}, x_j, x_{j-1} and x_{j+1}.
- *POS Between* – Part-of-speech tag of all tokens between x_i and x_j.
- *Feats* – Syntactic and morphological features that are included in the CoNLL-2006 dataset. We include these features for x_i, x_j and all tokens that occur between x_i and x_j.

Fernandes et al. (2010b) present text chunking and clause identification for the Bosque corpus, which comprises the Portuguese CoNLL-2006 dataset. We perform additional experiments using basic features based on this information. These features are the following:

– *Chunk Tag* – The chunk tag, in IOB2 style, for x_i and x_j.

– *Start Clause* – Indicates whether a token starts a clause.

– *End Clause* – Indicates whether a token ends a clause.

6.5 Empirical Results

The CoNLL-2006 (Buchholz and Marsi, 2006) provided a dependency parsing dataset that is derived from Bosque (Freitas et al., 2008), a Portuguese corpus comprising European and Brazilian news articles. In Table 6.1, we provide basic statistics about this dataset.

	Sentences	**Tokens**
Train	8,546	207,000
Test	241	5,838

Table 6.1: Bosque dependency parsing dataset statistics.

In this section, we compare the performances of systems based on the proposed ESL framework with state-of-the-art systems. Our systems use only first-order features, while the best performing systems for this task use second- and third-order features. In Table 6.2, we show the performances of several systems along with two systems based on ESL. The two first rows in the

System	**Learning Algorithm**	**Basic Features**	**Feature Generation**	**UAS**
Dual Decomposition	MIRA	3rd order	Manual	93.03
		2nd order		92.57
MSTParser	MIRA	2nd order	Manual	91.36
		1nd order		90.68
SPerc	SPerc	1st order	Manual	90.06
ESL	SPerc	1st order	EFG	90.28
		1st+ck+clause	EFG	92.66

Table 6.2: Performances of ESL and state-of-the-art systems on the Portuguese CoNLL-2006 dependency parsing dataset.

table present the system results by Koo et al. (2010). This system uses an algorithm based on dual decomposition (DD) to approximately solve the NP-hard optimization problem when second- and third-order features are considered. The DD algorithm provides a certificate of optimality for 99.65% of the test examples. The third and fourth rows in the results table show the results of MSTParser with first- and second-order features (McDonald et al., 2005, 2006; McDonald and Pereira, 2006). As we showed before, our ESL system outperforms an SPerc with the first-order templates from McDonald et al. (2006). When we provide text chunking and clause features (Fernandes

et al., 2010b) to our ESL system, we achieve a performance comparable with systems based on second- and third-order features. Moreover, this improvement is achieved by simply including two basic features, without any human effort, as would be required if one used manual templates.

We use 10% of the training data as validation set in order to pick the ESL meta-parameters. The loss weight C is set to 300 and the number of epochs is 10. We generate templates containing from 2 to 4 basic features.